

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Sistemas de Información de Gestión y Business Intelligence

Guía de Instalación, Configuración y Ejecución: LIVE

LIVE!

Javier Gómez Martínez



1. OBJETO

En la presente guía se detallan los pasos para instalar y configurar el software necesario para correr un servidor de la aplicación web *LIVE*, sistema recomendador de música en directo desarrollado para la asignatura Sistemas de Información de Gestión y Business Intelligence, perteneciente al Grado en Ingeniería Informática de la Universidad de León.



2. INSTALACIÓN

2.1. NODE.JS

LIVE emplea el entorno de ejecución **node.js** tanto en su *back-end*, utilizando el *framework Express.js*, como en el *front-end*. Es por ello un requisito contar con una instalación del mismo en la máquina servidora.

La versión más reciente de **node.js** puede descargarse desde la página oficial del proyecto¹. Allí se ofrecen 2 versiones. A lo largo del desarrollo de *LIVE*, se ha utilizado la rama 16.x, que posee soporte extendido (*LTS*, por sus siglas en inglés), por lo que es recomendable optar por esta opción.

Node.js® es un entorno de ejecución para JavaScript construido con V8, motor de JavaScript de Chrome.

New security releases to be made available January 10th, 2022

Descargar para Windows (x64)

16.13.1 LTS Recomendado para la mayoría	17.3.0 Actual Últimas características
---	---

[Otras Descargas](#) | [Cambios](#) | [Documentación de la API](#)

[Otras Descargas](#) | [Cambios](#) | [Documentación de la API](#)

O eche un vistazo al [Programa de soporte a largo plazo \(LTS\)](#)

Figura 1: página de descarga de Node.js

Tras la apertura del ejecutable, se inicia un asistente de instalación. Se recomienda mantener todas las opciones por defecto, y avanzar mediante sucesivos clicks en *Siguiente*.

¹ <https://nodejs.org/es/>



Welcome to the Node.js Setup Wizard



The Setup Wizard will install Node.js on your computer.

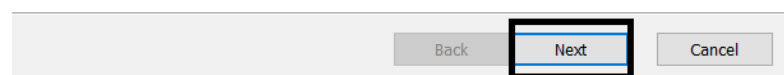


Figura 2: asistente de instalación de node.js

El único paso que puede demandar en mayor medida nuestra atención es aquel en el que el asistente sugerirá la instalación de varias herramientas necesarias para compilar algunos módulos del entorno. *LIVE* no ha necesitado de ninguno de estos módulos, por lo que no es necesaria la instalación de estas herramientas.

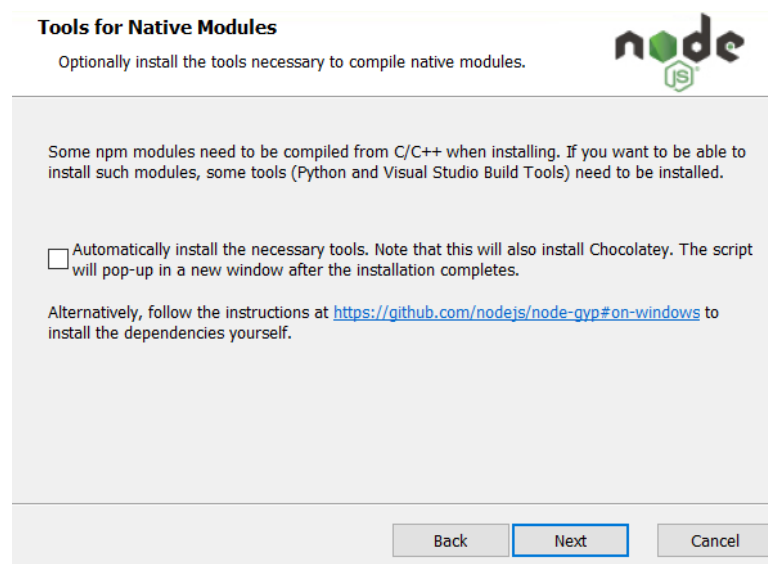


Figura 3: instalación de herramientas adicionales

Tras finalizar el asistente de instalación no se requiere ninguna acción adicional.



2.2. NEO4J

Como sistema de gestión de bases de datos (*DBMS*, por sus siglas en inglés), *LIVE* utiliza la plataforma basada en grafos **neo4j**, en la cual se almacena la información sobre la música y las preferencias del usuario.

Neo4j posee de varias ediciones, así como la posibilidad de elegir entre un servidor *on-premise* (como es el caso), o su plataforma en la nube, *Aura*. *LIVE* fue desarrollado utilizando la versión *Desktop*, cuya edición gratuita puede ser obtenida previo registro en la página oficial de la compañía².

De esta manera, en la página principal se optará por la opción “*Get Started*”, y se rellenará el formulario ofrecido después, el cual pide varios datos de contacto (nombre, apellidos, país, correo electrónico y organización).

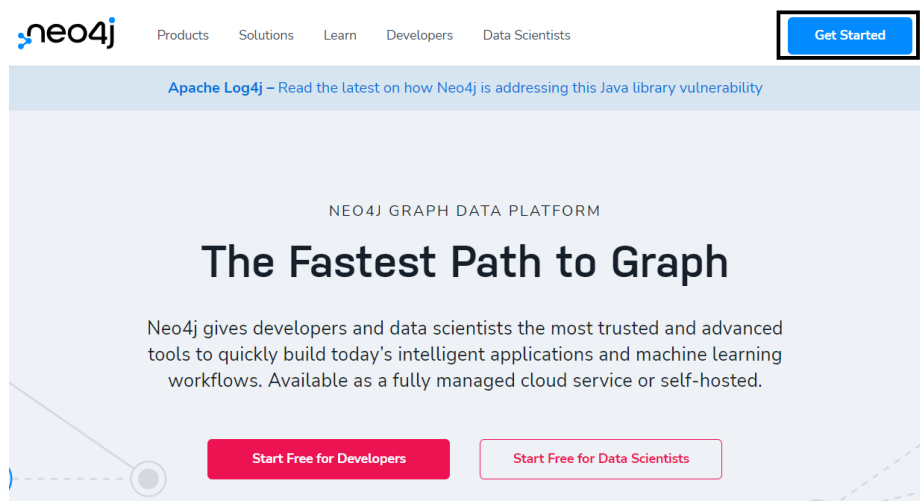
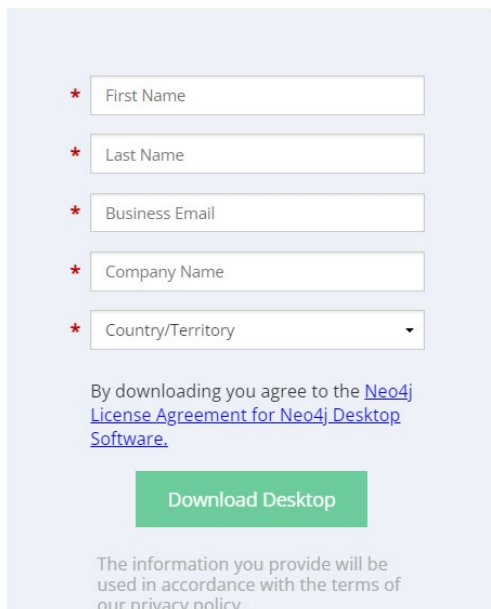


Figura 4: página web principal de neo4j

² <https://neo4j.com/>

Get Started Now

Please fill out this form to begin your download



A registration form with five fields, each preceded by a red asterisk: 'First Name', 'Last Name', 'Business Email', 'Company Name', and 'Country/Territory' (a dropdown menu). Below the fields is a link: 'By downloading you agree to the [Neo4j License Agreement for Neo4j Desktop Software.](#)'. A green button labeled 'Download Desktop' is positioned below the link. At the bottom, a small text block states: 'The information you provide will be used in accordance with the terms of our [privacy policy.](#)'

Figura 6: formulario de descarga

Después de enviar el formulario, se obtiene inmediatamente una clave de producto que será necesaria durante la instalación. Se aconseja copiarla al portapapeles mediante el botón que se ofrece encima. De forma paralela comenzará a descargarse el asistente de instalación.

Neo4j Desktop Activation Key

Use this key to activate your copy of Neo4j Desktop for use.

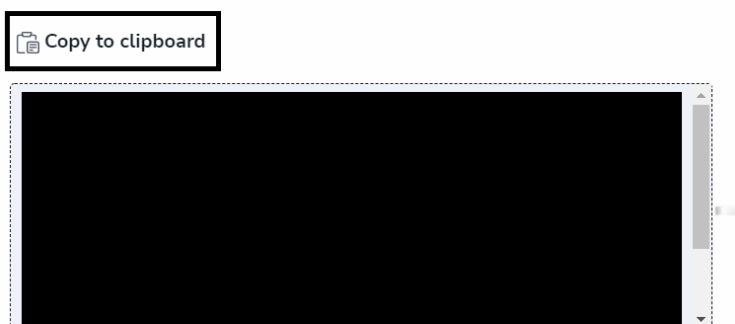


Figura 7: clave de activación de neo4j

Ya en el asistente de instalación, se recomienda dejar todas las opciones por defecto y avanzar haciendo *click* en siguiente. Es posible que a lo largo del proceso se muestre una alerta del cortafuegos, solicitando permisos para que *neo4j* se comunique en red. Dado que a priori no manejaremos información sensible, en este ejemplo se concederán permisos tanto para comunicar en redes públicas como privadas.

NOTA: los permisos se aplicarán a todas las bases de datos creadas en dicha instalación de *neo4j*, por lo cual se deberán revisar si en algún momento se utiliza el *DBMS* para otros usos e información.

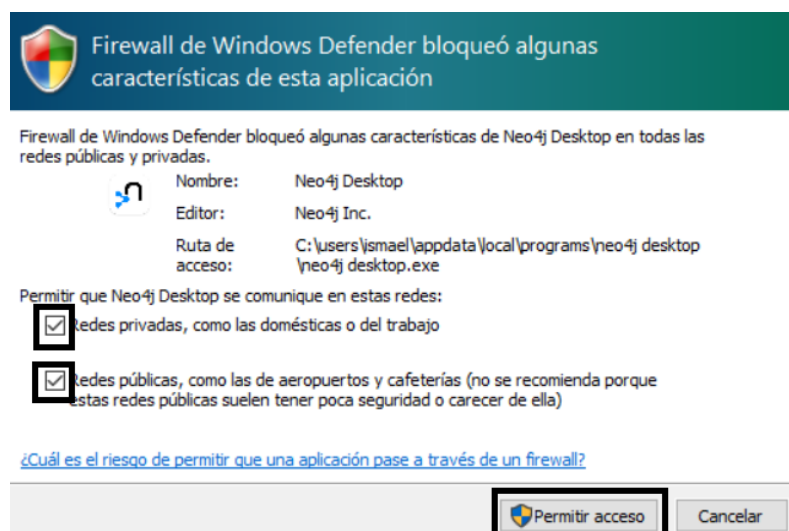


Figura 8: alerta del cortafuegos

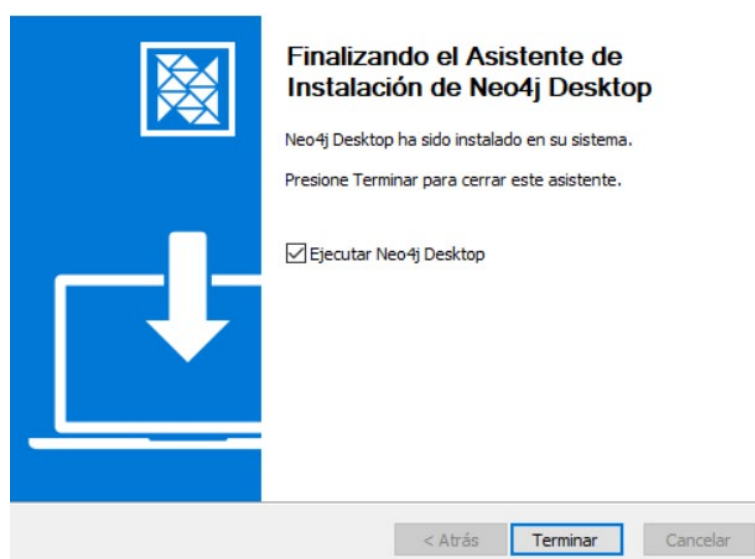


Figura 9: finalización del asistente de instalación



Una vez finalizada la instalación, se realizará una primera ejecución de *neo4j*, en la que se solicitará la clave de producto obtenida varios pasos atrás. No es necesario cumplimentar el formulario que se muestra en la misma ventana. Finalmente, si la clave es correcta, se mostrará el *dashboard* o ventana principal de *neo4j*.

Software registration

Neo4j Desktop is always free. Registration lets us know who has accepted this gift of graphs.

Register yourself with the following contact information.

Name *

Email *

Organization *

[Read about our privacy policy.](#)

OR

Already registered? Add your software key here to activate this installation.

Software key *

Software keys look like a long block of hexadecimal characters.

Register later

Activate

Figura 10: introducción de la clave de producto

3. CREACIÓN DE LA BASE DE DATOS

Una vez concluida la instalación de *neo4j*, en la página principal de la aplicación se mostrará un proyecto de ejemplo, una base de datos de películas (*Movie DBMS*). Ésta se habrá arrancado y estará dando servicio, por lo que en primer lugar se procederá a detenerla mediante el botón *Stop* situado en la parte superior derecha de la ventana.

Posteriormente, se creará el nuevo proyecto que albergará la base de datos de *LIVE*. Asegurándonos que en la barra lateral izquierda estamos en el apartado *Projects* (en caso contrario, podemos acceder a él pulsando en el primer botón por arriba de dicha barra), hacemos *click* en la opción *New* (3).

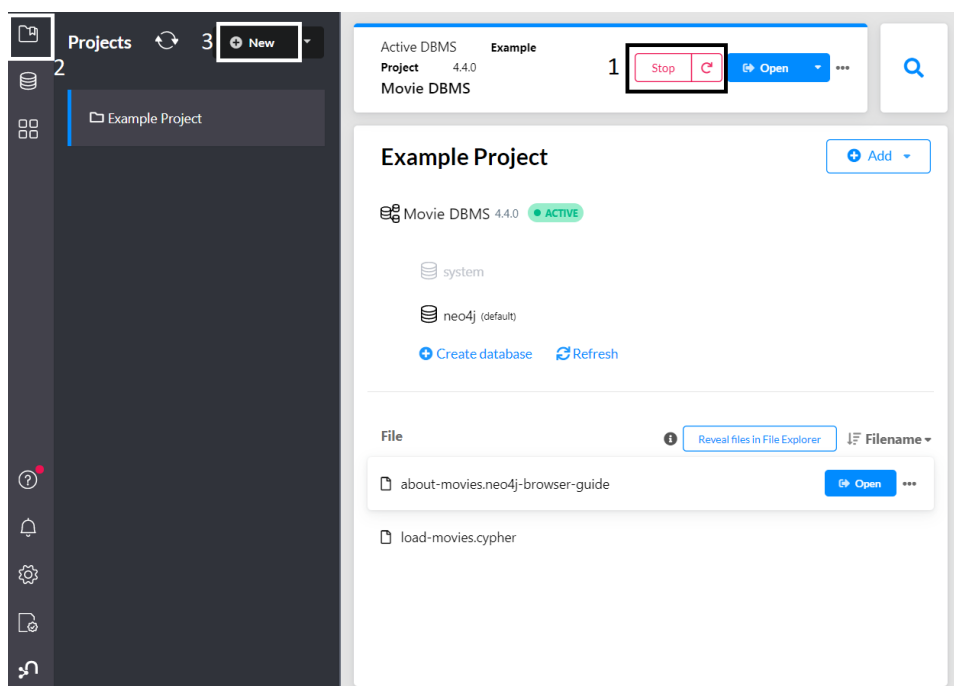


Figura 11: página principal con *Movie DBMS* en ejecución y los distintos botones remarcados

Sin más demora, se creará un nuevo proyecto de nombre *Project*, completamente vacío. Cada proyecto puede albergar una o más bases de datos. Solo será necesaria una, la cual procederemos a crear *clickando* en el botón *Add* situado a la derecha del nombre del proyecto. De entre las opciones desplegadas, seleccionaremos la opción *Local DBMS*.

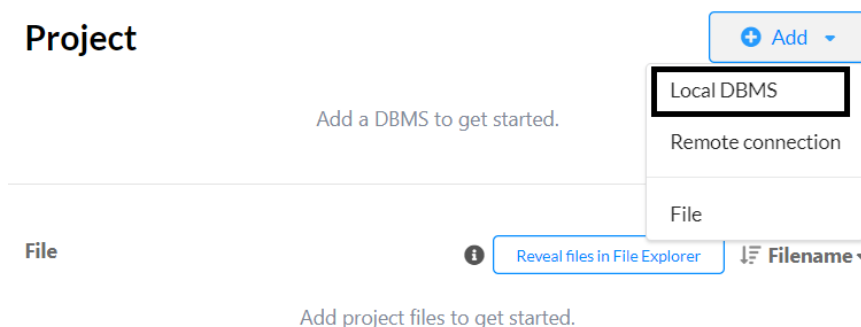


Figura 12: proyecto de nombre “Project” en blanco

Se abrirá un nuevo cuadro emergente, en el que se pedirá un nombre para esa base de datos y una contraseña. De nombre, es posible dejar el que se encuentra por defecto (*Graph DBMS*), no siendo relevante el empleo de otro. Sí lo será la contraseña, que, mientras no sea modificada en el *back-end*, debe ser “*SIBI_LIVE*”, sin comillas. Por último, es posible seleccionar la versión de *neo4j* a emplear. *LIVE* ha sido desarrollado utilizando la rama 4.3.x, aunque, como se puede comprobar en la siguiente figura, en este ejemplo se desplegó sobre una versión 4.4.0, sin incidencias conocidas. En cualquier caso, para asegurar el correcto funcionamiento de la aplicación, es recomendable la rama 4.3.x.

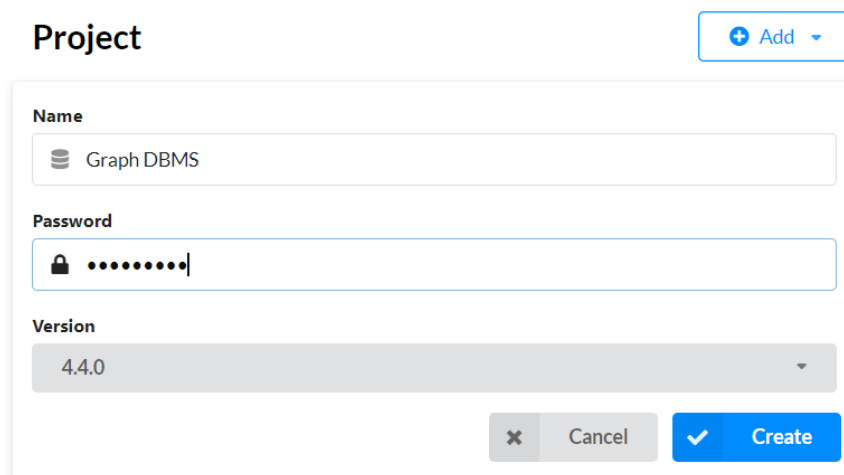


Figura 13: creación de la base de datos Graph DBMS

Una vez creada, procederemos a arrancar la base de datos mediante el botón *Start*, situado donde antes estaba el botón *Stop*. Transcurrido un tiempo cercano a un minuto, la BBDD estará ofreciendo servicio, aunque, lógicamente, no tendrá contenido.

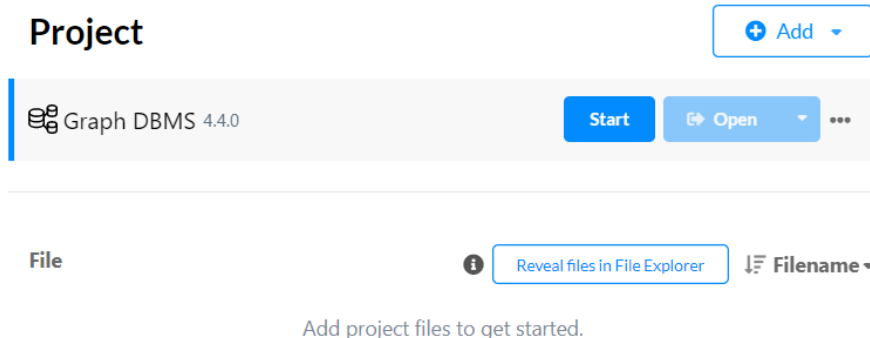


Figura 14: base de datos Graph DBMS creada

A continuación se procederá a dotar de contenido a la base de datos. Para ello serán necesarios dos archivos, *cancionesDef.csv*, que lista los distintos nodos de tipo *Song*, y *relacionesDef.csv*, con las distintas relaciones entre ellas. Ambos archivos se encuentran dentro de la carpeta *Datasets* del repositorio de *LIVE* en GitHub³. Si aún no han sido descargados, se recomienda descargar el archivo comprimido de todo el contenido del repositorio. Éste puede ser descomprimido a posteriori en cualquier ubicación, por ejemplo, en la raíz del disco C.

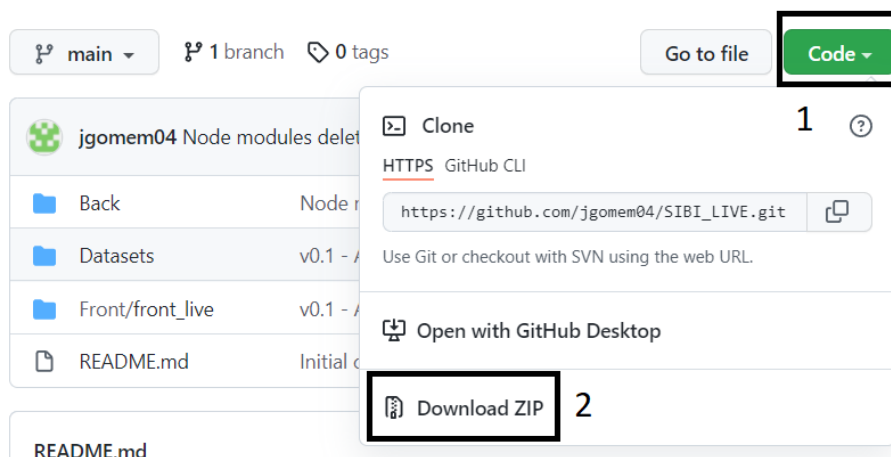


Figura 15: descarga del contenido del repositorio desde GitHub

Para importar los archivos *csv*, se regresa a *neo4j Desktop*, y, desplegando el menú contextual localizado a la derecha del botón *Open*, se hace click en la opción *Open folder -> Import*. Se abrirá la ubicación donde *neo4j* localiza los ficheros que se importarán a la base de datos.

³ https://github.com/jgomem04/SIBI_LIVE

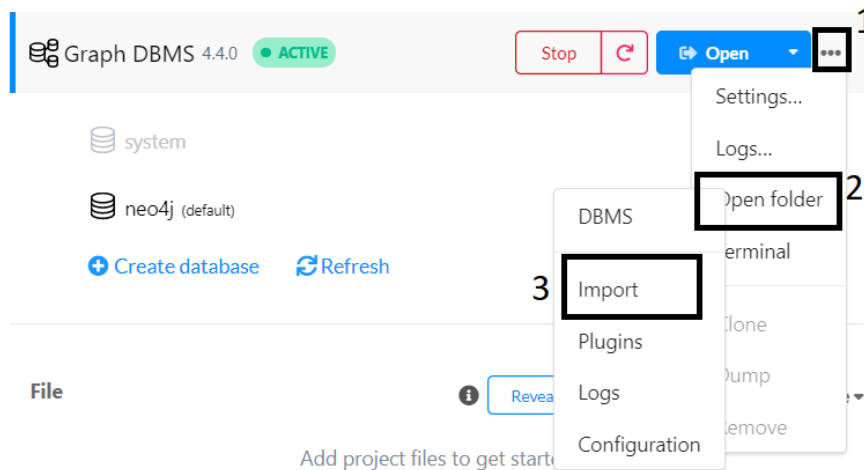


Figura 16: localización de la opción *Open folder* -> *Import*

A continuación, en otra ventana separada, acudiremos al directorio donde se ha descomprimido el repositorio de *GitHub*, y se copiarán los dos archivos csv, pegándolos en la recién abierta carpeta de importación.

Solo resta que *neo4j* lea los archivos, transformando su contenido en el correspondiente grafo. Esta tarea se realizará desde *neo4j Browser*, accesible desde el botón *Open* que también es observable en la Figura 16.

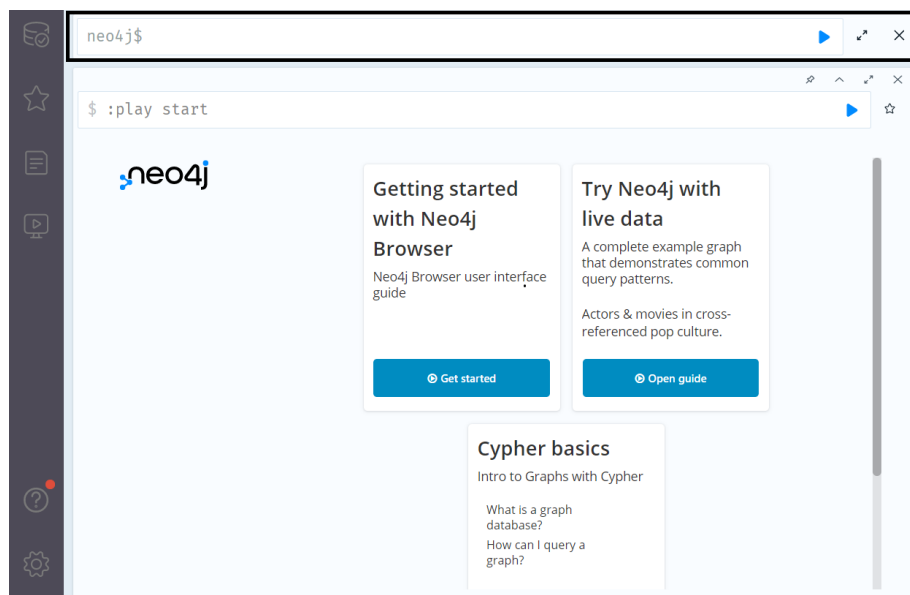


Figura 17: página principal de *neo4j Browser*



En la parte superior de la interfaz se puede observar el equivalente al *prompt*, donde introduciremos los distintos comandos. Comenzaremos creando las *constraints*, que nos aseguran que no existe información duplicada en la base de datos. Concretamente, crearemos *constraints* para los usuarios y las canciones:

```
CREATE CONSTRAINT EmailIsUnique on (p:Person) ASSERT p.email  
IS UNIQUE  
CREATE CONSTRAINT UsernameIsUnique on (p:Person) ASSERT p.user  
IS UNIQUE  
CREATE CONSTRAINT SongIdIsUnique ON (s:Song) ASSERT s.id IS  
UNIQUE
```

Creadas las *constraints*, se procede a la lectura del fichero de canciones:

```
LOAD CSV WITH HEADERS FROM 'file:///cancionesDef.csv' AS row  
MERGE (s:Song {id: toInteger(row.id)})  
ON CREATE SET  
    s.title = row.title,  
    s.artist = row.artist,  
    s.bpm = toFloat(row.bpm),  
    s.energy = toFloat(row.energy),  
    s.idSpotify = row.idSpotify,  
    s.genre = row.genre,  
    s.preview = row.preview,  
    s.cover = row.cover,  
    s.date = row.date
```

Finalmente, se lee el fichero de relaciones:

```
LOAD CSV WITH HEADERS FROM 'file:///relacionesDef.csv' AS row  
MATCH (s:Song {id: toInteger(row.idA)}) MATCH  
(t:Song {id: toInteger(row.idB)})  
MERGE (s)-[:RELATED]-(t)
```

Al finalizar la lectura de ambos ficheros, es posible asegurar que el proceso se ha realizado correctamente si se obtienen los mensajes que se muestran en las siguientes figuras.



Added 3913 labels, created 3913 nodes, set 39130 properties, completed after 544 ms.

Figura 18: mensaje tras la importación de las canciones

Created 19203 relationships, completed after 49498 ms.

Figura 19: mensaje tras la importación de las relaciones

Dentro del funcionamiento del sistema recomendador, en ocasiones se realizará una ordenación de las canciones por *PageRank* personalizado. Para ello, *neo4j* utiliza lo que se conoce como un grafo catalogado, es decir, un grafo almacenado en memoria principal que permite realizar búsquedas a mayor velocidad.

Dicho grafo catalogado, así como otros de los algoritmos empleados por *LIVE*, requieren de la instalación de las librerías *APOC* y *Graph Data Science Library (GDS)*. La incorporación de las mismas puede realizarse desde la pantalla principal de *neo4j Desktop*, seleccionando el recién creado *Project* y acudiendo al apartado *Plugins*, como se puede observar en la siguiente figura.

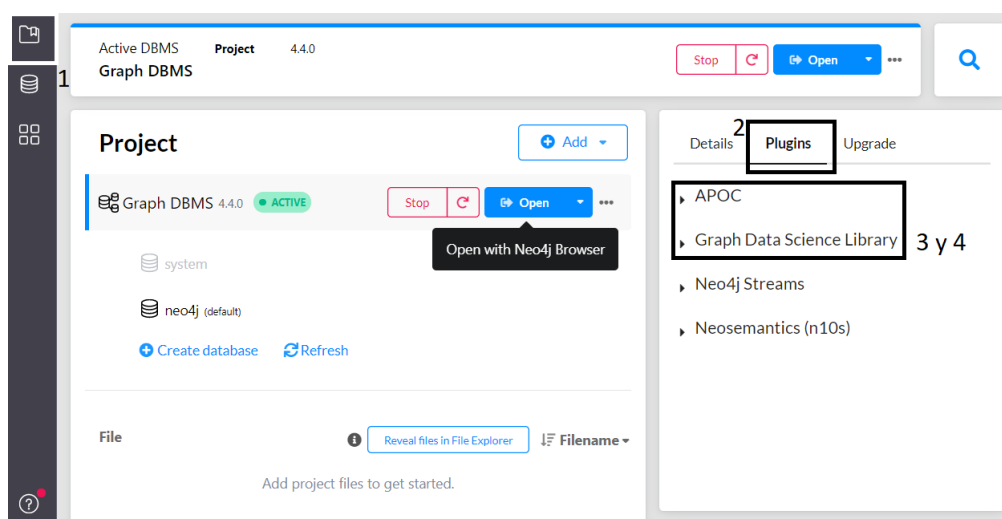


Figura 20: instalación de plugins en neo4j Desktop

La instalación realizará automáticamente un rearranque de la base de datos. Finalizado éste, se puede regresar a *neo4j Browser* y crear el grafo catalogado:



```
CALL gds.graph.create.cypher(  
  'grafoCanciones',  
  'MATCH (s:Song) RETURN id(s) as id',  
  'MATCH (s:Song)-[:RELATED]-(t:Song) RETURN id(s) AS  
    source, id(t)  
    AS target')  
YIELD graphName AS graph, nodeQuery, nodeCount AS nodes,  
  relationshipQuery, relationshipCount AS rels
```

La operación se habrá realizado correctamente si se observa un mensaje como el siguiente:

The screenshot shows the Neo4j Cypher query interface. At the top, the query is entered: `neo4j$ CALL gds.graph.create.cypher('grafoCanciones', 'MATCH (s:Song) RETU...`. Below the query, a table displays the results. The table has four columns: `graph`, `nodeQuery`, `nodes`, and `relationshipQuery`. The first row shows the results for the 'grafoCanciones' graph, with 3913 nodes and a relationship query. The status bar at the bottom indicates: 'Started streaming 1 records after 11 ms and completed after 2637 ms.'

	graph	nodeQuery	nodes	relationshipQuery
1	"grafoCanciones"	"MATCH (s:Song) RETURN id(s) as id"	3913	"MATCH (s:Song)-[:RELATED]-(t:Song) RETURN id(s)"

Started streaming 1 records after 11 ms and completed after 2637 ms.

Figura 21: grafo catalogado "grafoCanciones" creado correctamente

NOTA: como bien se ha mencionado, un grafo catalogado se almacena en memoria principal, por lo que será eliminado al finalizar la instancia de *neo4j*. Por cada rearranque del *DBMS*, se debe volver a crear el grafo catalogado para contar con toda la funcionalidad de *LIVE*.

4. ARRANQUE DEL BACK-END

Finalizadas todas las configuraciones, es hora de arrancar el servidor *back-end*. Para ello, es necesario abrir una terminal o consola de comandos, y dirigirse al directorio donde está ubicado el punto de arranque *Express.js*, es decir, la carpeta *Back* del repositorio de GitHub, el cual ha sido descargado en su totalidad y descomprimido en la ubicación de preferencia del usuario. En este ejemplo, como se mencionó anteriormente, se descomprimió en la raíz del disco C, dando lugar a una ruta hasta *Back* como la que se puede observar a continuación.

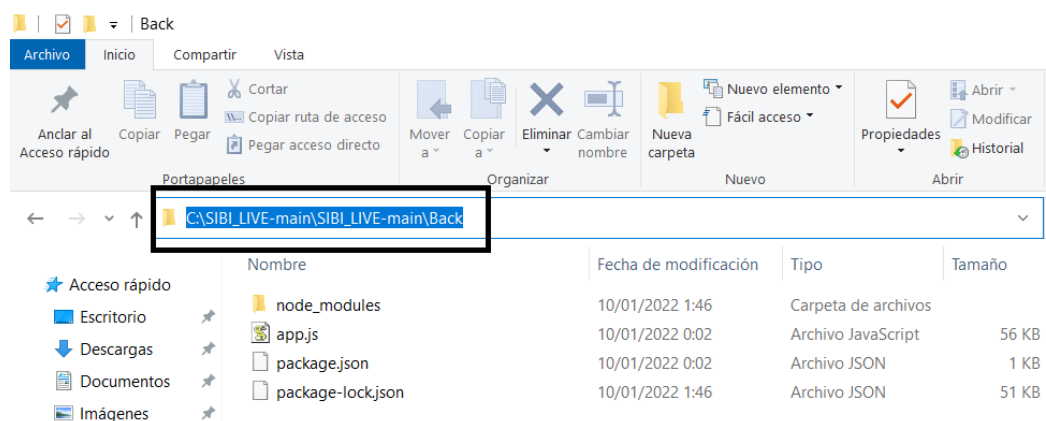


Figura 22: ruta de la carpeta *Back*

Para navegar desde la consola de comandos hasta el directorio deseado, bastará con utilizar el comando `cd`, seguido de la ruta copiada del explorador de archivos. Siguiendo el ejemplo:

```
cd C:\SIBI_LIVE-main\SIBI_LIVE-main\Back
```

Para arrancar el servidor *Express* se utilizará la herramienta *nodemon*, que permite detectar cualquier cambio realizado en los ficheros y reiniciar automáticamente el servidor para aplicar dichas modificaciones. *Nodemon* puede instalarse mediante el gestor de paquetes de *node.js*, *npm*, utilizando el comando:

```
npm install -g nodemon
```

Donde `-g` indica que deseamos realizar una instalación global, es decir, que no será necesario volver a instalar *nodemon* en otros proyectos que podamos desplegar en el futuro.



Previo al arranque del servidor, es necesario descargar las dependencias del proyecto. Se utilizará de nuevo *npm* y su comando:

```
npm install
```

Finalizada la instalación, para arrancar el servidor bastará con ejecutar el comando:

```
nodemon app.js
```

Si se muestra el mensaje “*Backend escuchando en el puerto 3000*”, el proceso se habrá realizado correctamente y el servidor *Express* estará operativo.

```
C:\SIBI_LIVE-main\SIBI_LIVE-main\Back>npm install -g nodemon
added 116 packages, and audited 117 packages in 15s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\SIBI_LIVE-main\SIBI_LIVE-main\Back>nodemon app.js
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Backend escuchando en el puerto 3000
```

Figura 23: backend operativo

5. ARRANQUE DEL FRONT-END

Arrancado el *back-end*, solo resta arrancar el correspondiente *front*. De la misma forma que se realizó en el punto 4, se acudirá al correspondiente directorio, en este caso, *Front/front_live*.

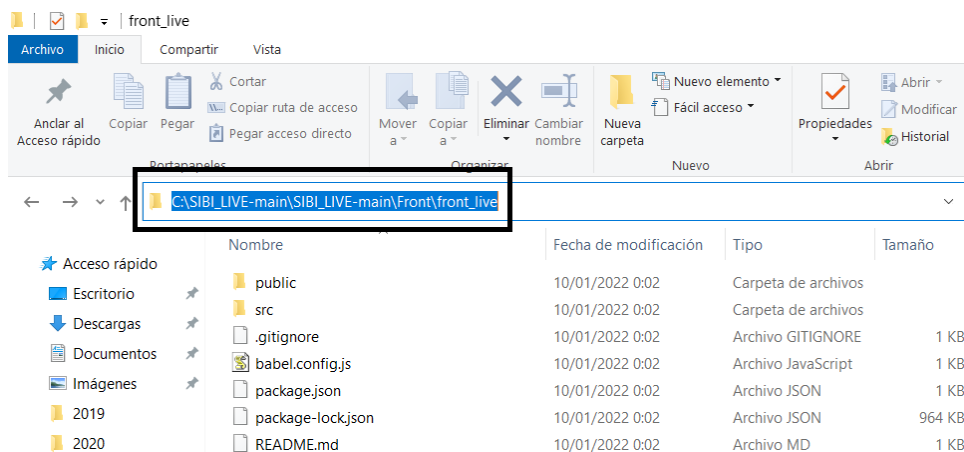


Figura 24: ruta de la carpeta *Front/front_live*

Siguiendo el ejemplo, en una nueva terminal, para desplazarse al directorio deseado, se utilizaría el comando:

```
cd C:\SIBI_LIVE-main\SIBI_LIVE-main\Front\front_live
```

De nuevo, se instalarán las dependencias, en esta ocasión del front, utilizando:

```
npm install
```

Y finalmente, para compilar todos los paquetes necesarios y arrancar el servidor *front-end*, se utiliza el comando:

```
npm run serve
```

Tras la compilación se mostrará un enlace desde el cual acceder a la interfaz de *LIVE*.



```
DONE Compiled successfully in 4850ms

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.0.10:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Figura 25: front-end compilado y ejecutándose

Accediendo al enlace *localhost* desde cualquier navegador web podremos observar la página de inicio de sesión en *LIVE*, pudiéndose a partir de entonces registrar un nuevo usuario y comenzar a utilizar las distintas características de la aplicación.

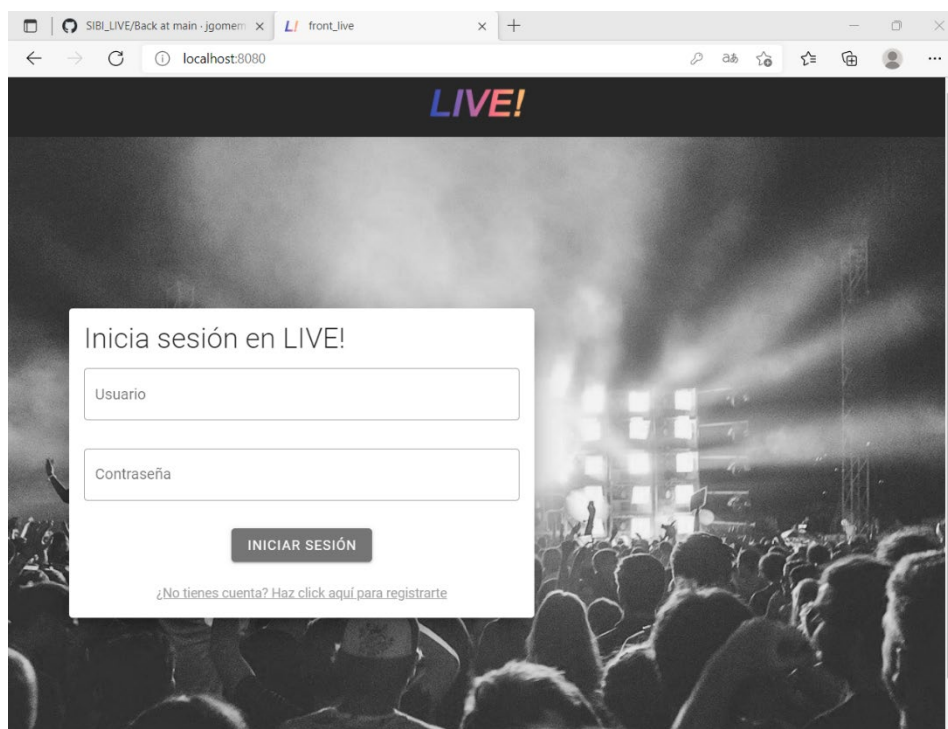


Figura 26: página de inicio de sesión de LIVE